3
Aprendizado supervisionado

Neste capítulo será apresentado as técnicas para aprendizagem adotadas e como o sistema foi modelado para suportá-las.

3.1.Técnicas para aprendizado supervisionado

Pesquisando na literatura, encontramos muitos algoritmos para aprendizado supervisionado. Cada um utiliza uma técnica diferente para determinar a função aprendizado e adéqua-se melhor de acordo com o tipo de problema.

Embora a técnica utilizada não seja a mesma em todos os algoritmos citados aqui, todos precisam que o objeto a ser classificado, no caso o comentário, seja representado através de um vetor multidimensional chamado de *feature vector*, ou simplesmente *feature* ou, como iremos denominar, *atributo*. Na Seção 3.2 será detalhado os métodos de extração de atributos que utilizamos neste trabalho.

3.1.1.Naïve Bayes

Naïve Bayes é um classificador probabilístico baseado no Teorema de Bayes, que mostra como determinar a probabilidade de um evento condicional através da probabilidade inversa. Para facilitar a computação, este classificador assume que a presença (ou ausência) de um atributo não tem relação alguma com qualquer outro atributo (por isto o nome Naive).

Partindo do teorema de Bayes:

$$P(classe|atributos) = \frac{P(classe) \cdot P(atributos|classe)}{P(atributos)}$$

Como é assumido que os atributos $(a_1,..a_n)$ são independentes:

$$P(classe|atributos) = \frac{P(classe) \cdot P(a_1|classe) \cdot ... \cdot P(a_n|classe)}{P(atributos)}$$

Em vez de calcular $P(atributos)$ explicitamente, o algoritmo calcula apenas o denominador para cada classe e normaliza-os para que a soma seja 1. A

este termo damos o nome de evidência.

$$P(classe|atributos) = \frac{P(classe) \cdot P(a_1|classe) \cdot \ldots \cdot P(a_n|classe)}{evidência}$$

Se estivermos interessado apenas em qual classe tem maior probabilidade, a evidência pode ser ignorada pois é uma constante, determinada durante o treino.

Para a classificação binária sabemos que a probabilidade das classes, para os mesmos atributos, são complementares, assim precisamos apenas ter a probabilidade do comentário ser rejeitado.

$$P(aprovado|atributos) = 1 - P(rejeitado|atributos)$$

Quando a probabilidade de rejeição for menor que 0.5 o comentário é classificado como aprovado. Quando for maior que 0.5 é classificado como rejeitado. Quando a probabilidade é igual a 0.5, definimos que ele será rejeitado.

Para este trabalho estamos utilizando a implementação do classificador Naïve Bayes do toolkit NLTK (BIRD e LOPER, 2011). Embora esta biblioteca não tenha a implementação mais performática, ela oferece uma boa documentação e é implementada em Python, que é a mesma linguagem no qual este trabalho será implementado.

3.1.2.Boostexter

O Boostexter é uma implementação de um algoritmo de classificação de texto baseado da técnica de *boosting*, em que diversos classificadores mais simples e de baixa acurácia são combinados em um único classificador com alta acurácia. Mais especificamente, para o *Adaboost*, os classificadores são treinados sequencialmente, sendo que cada classificador subsequente é treinado em exemplos que foram mais difíceis de classificar pelos classificadores anteriores (SCHAPIRE e SINGER, 2000).

O Boostexter pode ser implementado tanto sobre o Adaboost.MH quanto o Adaboost.MR, ambos projetados para classificação de dados que podem ser classificados simultaneamente em várias classes. A principal diferença entre o Adaboost.MH e o Adaboost.MR está no fato do primeiro trabalhar minimizando a perda de *Hamming* enquanto o segundo foi projetado para encontrar uma hipótese que ranqueia as classes de maneira a garantir que as corretas sempre estejam no topo da lista (SCHAPIRE e SINGER, 2000).

Muitas implementações de Adaboost utilizam redes neurais e árvore de decisão como classificadores internos (que serão combinados). Na implementação para classificação de texto é utilizado como classificador interno *stumps* (árvores de decisão de um único nível), em que a raiz avalia a presença ou ausência de um atributo no texto.

Para teste trabalho foi utilizado uma implementação de código aberto do Boostexter chamada Icsiboost (FAVRE e HAKKANI, 2011).

3.1.3.SVM

A teoria sobre o Support Vector Machine (SVM) foi proposta inicialmente por (VAPNIK e LERNER, 1963), como um método eficiente para a identificação de padrões. Ela possui muitas vantagens na resolução de problemas com uma amostra pequena, não-lineares e com um grande número de dimensões. (JOACHIMS, 1998) foi o primeiro a introduzi-lo para a classificação textual.

Ela é principalmente baseada nas seguintes considerações (YONG-FENG e YAN-PING, 2004):

- Minimizar o risco estrutural, minimizando a dimensão VC (Vapnik-Chervonenkis) de um conjunto mínimo de funções para controlar o risco estrutural do aprendizado de máquina, de modo a torna-lo eficiente.
- Maximizar a distância entre classes (encontrar o hiperplano de classificação ótimo) para realizar o controle da dimensão VC, que pode ser garantido por um teorema da estatística.
- Utilizar funções kernel para trabalhar num espaço de maior dimensão, porém linear.

Comparado com os tradicionais métodos de classificação, o SVM tem a vantagem de evitar overfitting e oferece boa performance para classificação de texto, o que também depende do tipo de kernel escolhido. Aliás o critério de escolha do tipo de kernel e seus parâmetros são um ponto fraco do algoritmo, pois ainda são muito baseados em experiências humanas.

Para este trabalho foi escolhido a implementação do SVM de (FAN, CHANG, *et al.*, 2008), chamada de *LIBLINEAR* (nome devido a função kernel utilizada ser linear). Ela é uma versão otimizada especialmente para problemas de

classificação de documento onde o número de atributos é maior que o número de exemplos de treino.

Uma outra limitação do SVM é o fato de apenas suportar atributos numéricos. Assim, para converter a representação de texto em vetores, a cada palavra do texto é atribuído um índice numérico sequencial e para cada coordenada do vetor é atribuído o valor 1 se a palavra está presente na instância, ou 0 caso contrário. Como o número de palavras em uma instância é sempre muito menor que o total de palavras existentes, a matriz que representa a presença ou ausência de uma palavra no texto tende a ser esparsa, não sendo necessário representar os atributos onde ela é zero para cada instância.

3.1.4.Maximum Entropy

Modelos de Entropia Máxima são modelos exponenciais que implementam a intuição de que, se não existe nenhuma evidência que favoreça uma alternativa de solução em relação a uma outra, então ambas as alternativas devem ser consideradas com probabilidades iguais (KOELING, 2000). De forma mais objetiva, o princípio da entropia máxima afirma que a distribuição $p(a,b)$ mais correta é aquela que maximiza a entropia, ou incerteza, e que respeita determinadas restrições que representam as evidências (os fatos) conhecidas para aquele experimento. Ou seja, se A denota o conjunto de possíveis classes e B denota o conjunto de possíveis atributos, a melhor distribuição p, consistente com as "informações parciais" existentes, é a que maximiza a entropia

$$H(p) = -\sum_{x \in S} p(x) log\ p(x)$$

onde $x = (a,b), a \in A, b \in B$ e $S = A \times B$ (RATNAPARKHI, 1997).

Para a estimação destes parâmetros, geralmente é usado o algoritmo Improved Iterative Scaling (IIS) (SKUT e BRANTS, 1998), no qual é assumido que p tem a forma:

$$p(a|b) = \frac{1}{Z(b)} \cdot e^{\Sigma_i\ \lambda_i f_i(a,b)}$$

onde $f_i(a,b)$ é uma função de valor binário e que verifica, em (a,b), uma característica de interesse para o problema. λ_i é um parâmetro que indica o quanto f_i é importante para o modelo e $Z(b)$ é um fator de normalização.

Uma diferença importante entre o Naive Bayes e o modelo de entropia, é que este último não precisa assumir a independência entre os atributos. Em contrapartida, o cálculo de $p(a|b)$ é computacionalmente mais caro que o Naive Bayes, tornando-o mais ineficiente.

3.2.Métodos de extração de atributos

Para que os classificadores citados acima consigam determinar uma função de aprendizado é necessário que os comentários sejam convertidos em vetores, selecionados de maneira a fornecer o máximo de informação sobre a probabilidade de rejeição do comentário. Estes vetores são chamados de *feature vectors*, ou simplesmente *features* ou atributos.

A escolha de uma função responsável por converter os comentários em vetores é fundamental para o bom desempenho de quaisquer um dos algoritmos utilizados neste trabalho, pois ela que irá conter as partes do comentário que irá relacioná-lo com a probabilidade de ser reprovado ou aprovado.

3.2.1.Bag of Words

Um dos métodos de extração de atributos mais simples e mais utilizado em problemas de classificação de texto é o saco de palavras (*bag of words*). Neste método, cada palavra é um atributo do texto. Neste tipo de extração somente palavras e números são utilizados como atributo, ignorando todos os sinais de pontuação. Uma variação deste método é quando os caracteres de pontuação também são incluídos.

Devido a grande quantidade de erros de ortografia presente no corpus, as palavras são convertidas para minúscula e todos os acentos são removidos. Isto evita que “não” e “nao” sejam considerados como atributos diferentes, porém faz “vovô” e “vovó” serem considerados como mesmo atributo.

3.2.2.Corretor

Analisando os corpora utilizados neste trabalho, percebemos que muitas palavras possuem erros de ortografia. No caso do corpus globo-comments (comentários de notícias), são comuns erros devidos à digitação. Já para o corpus

globo-twitter muitos erros de ortografia são parte do estilo de escrita deste tipo de rede social (como “naum” em vez de “não”). Certamente estes erros de ortografia têm impacto no classificador, pois geram atributos diferentes para a mesma palavra. Para evitar isto, construímos um método de extração de atributos que utiliza sempre uma versão corrigida da palavra como atributo ou a própria, caso já estivesse correta de acordo com um dicionário.

Uma heurística foi então construída para a correção das palavras, baseada no estudo do corpus. Inicialmente a palavra é verificada em 2 dicionários: de português e de inglês. Caso ela esteja em algum deles está correta. Caso não esteja, os métodos abaixo são aplicados até que se consiga obter uma palavra ortograficamente correta.

1. Tenta-se retirar os caracteres repetidos das palavras um a um e, para cada nova palavra gerada, sua existência é confirmada nos dicionários. A verificação a cada palavra repetida é necessária para evitar que a palavra, ortograficamente errada, “carrro” após a correção vire a palavra “caro”.
2. Palavras terminadas com “aum” ganham o sufixo “ão” e novamente o dicionário é verificado.
3. Expressões comuns como “haha...”, “rsrs...” e “auau....” são alteradas para “ha”, “rs” ou “au” independente da quantidade de repetições existentes ou de iniciarem com a segunda letra e/ou terminarem com a primeira (como “ahahah”).
4. Se a palavra não existe no dicionário mas foi redigida com inicial maiúscula e todas as outras minúsculas, então deve ser considerada um nome próprio e assumimos estar correta. Por exemplo, “Google” ou “Microsoft”.
5. Caso nenhuma das técnicas acima tenha corrigido a palavra então buscamos sugestões no dicionário de português. Os seguintes critérios são então utilizados:
    a. Caso nenhuma sugestão seja fornecida, a própria palavra é retornada. Neste caso estamos assumindo que é uma nova palavra ortograficamente correta.

b. Se existe uma palavra na sugestão que difere da palavra escrita apenas pelos caracteres de acentuação, esta será a correção
c. Por fim, é assumida como grafia correta da palavra aquela que possui a menor distancia de edição, calculada de acordo com (LEVENSHTEIN, 1966).

Os dicionários de português e inglês utilizados foram da biblioteca Aspell, na versão 0.60.6. O algoritmo para geração de sugestão utilizado foi o provido pelo próprio dicionário, e que reporta uma acurácia de 89,7% para sugestões em inglês. Os resultados detalhados e o corpus utilizado estão publicados em (GNU ASPELL, 2011).

3.2.3.POS Tagging

Outro método muito utilizado para geração de atributos para a classificação de texto é o uso de POS Tagging, onde distinguimos as palavras também pela sua classe gramatical. Por exemplo, veja os 2 comentários abaixo e observem o uso da palavra "porcos":

(1) "*Se um cão produz isso, imagina a quantidade de animais, entre* ***porcos****, bois, galinhas...*" (aprovado)

(2) "*Ficam esperando esses shows porcos ai e ...*" (reprovado)

No segundo exemplo, o comentário foi reprovado devido à existência da palavra "porcos" como adjetivo. No primeiro exemplo, a palavra "porcos" é um substantivo e, por isto, não fornece um indicativo de reprovação.

Como o texto possui muitos erros de ortografia, que prejudicariam a eficiência do classificador gramatical, antes do comentário ser classificado gramaticalmente, suas palavras são corrigidas pelo corretor ortográfico.

Foram utilizadas 2 implementações de POS Tagging para este trabalho:
Uma treinada com o classificador Naive Bayes e outra utilizando um classificador sequencial baseado apenas em memorizar as ocorrências mais comuns, como: (art,"décimo") → adj, ou seja, a palavra "décimo" após um artigo é classificada como adjetivo. Apesar das diferenças entre os algoritmos de classificação, a acurácia de ambos é muito parecida, em torno de 87% para o corpus Mac Morpho (ALUÍSIO, PELIZZONI, *et al.*, 2003). O que os diferencia é o tempo de

classificação, superior no classificador sequencial.

3.2.4.N-grams

Este não é propriamente um método para a geração de atributos, mas sim uma forma de combinar os atributos gerados pelos outros métodos, agrupando-os dois-a-dois (bigrams) ou três-a-três (trigrams). Somente são combinados os atributos vizinhos. Para entender melhor, veja o exemplo abaixo:

"*Assisti a matéria do Jornal Hoje do segundo dia...*"

Para este exemplo a combinação de 2 n-grams (bigrams) seria: "Assisti a"; "a matéria"; "matéria do"; "do Jornal"; "Jornal Hoje"; "Hoje do"; "do segundo"; "segundo dia".

Para evitar uma combinação muito grande entre atributos, o que aumentaria o tempo de classificação e poderia causar overfitting, limitamos o número de n-grams a tamanho 3 (trigrams). Ainda, iremos nos referir daqui a diante por bigrams sempre a combinação de unigrams + bigrams, o mesmo para trigrams que é uma combinação de unigrams + bigrams + trigrams. Unigrams é quando os atributos não são combinados entre si.

A vantagem de utilizar n-grams maior que 1 é permitir que nomes compostos possam ser identificados como um único atributo. No exemplo acima, "Jornal Hoje" seria considerado um único atributo.